# Lecture 2

# Graphics Programming

# Outline

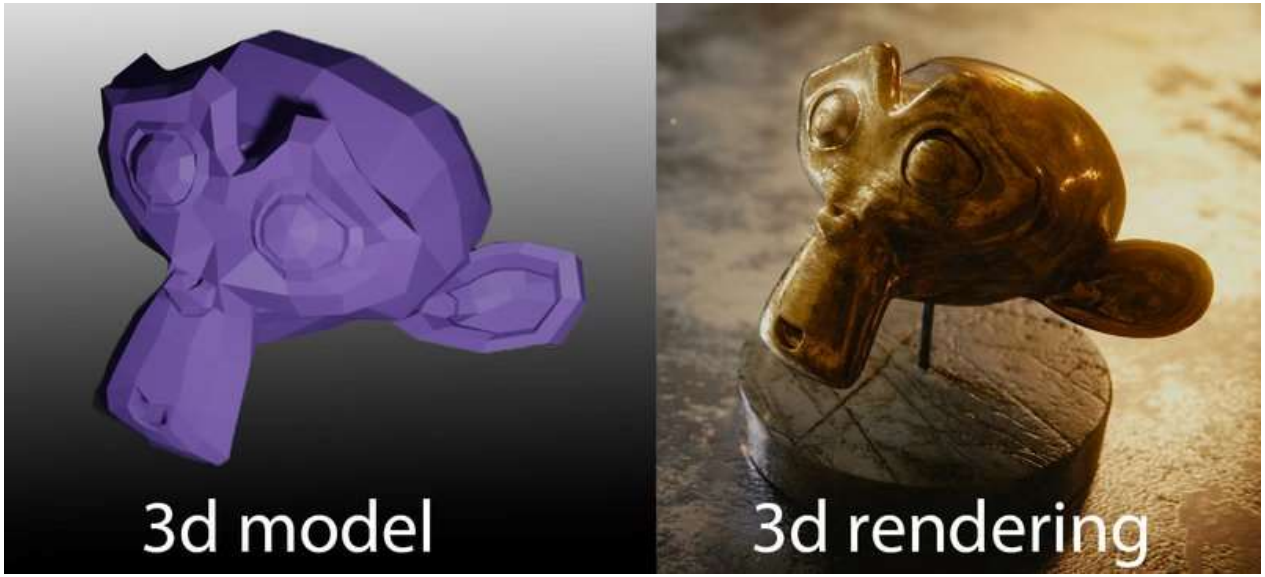
- What is Graphics Programming?
- Basics of OpenGL
- Setting Up OpenGL with GLUT
- OpenGL Programming Basics with GLUT
- Intermediate OpenGL Concepts

# What is Graphics Programming?

It is the process of creating visual content using computer hardware and software. It involves managing how shapes, colors, textures, and animations appear on the screen.

**What is Rendering?**

- The process of generating an image from a 2D/3D model by simulating lighting, textures, and geometry.



Images sources:
RealSpace 3D. (n.d.). *What is 3D rendering?* Retrieved from https://www.realspace3d.com/resources/what-is-3d-rendering/ (Accessed: 18 January 2025)
NoTriangle Studio. (n.d.). *3D Rendering vs. 3D Modeling*. Retrieved from https://notrianglestudio.com/3d-rendering-vs-3d-modeling/ (Accessed: 18 January 2025)

# Why Learn Graphics Programming?

**Why Graphics Programming is important:**

- **Interactivity:** Create engaging and responsive visual content.
- **Creativity:** Express ideas and concepts visually through software.
- **Career Opportunities:** High demand in gaming, film, virtual reality, and simulation industries.

**Core Skills Gained:**

- Understanding of math concepts (linear algebra, geometry).
- Familiarity with rendering pipelines and GPU programming.
- Problem-solving and optimization techniques for real-time applications.

# What Is OpenGL?
## (Open Graphics Library)

It is a cross-platform API, open standard API for rendering 2D and 3D graphics.

**OpenGL is an API, not a Graphics Engine**

Unlike engines (Unity, Unreal), OpenGL provides the tools for developers to build graphics from scratch, giving complete control.

OpenGL functions are in a single library named ***GL*** (or OpenGL). It contains 120 core drawing functions for 2D primitives. All functions starts with "***gl***" for example : `glVertex2D(x,y), glFlush()`

# Why Use OpenGL?

1. Easy to learn for beginners. Allow you to build skills transferable to other graphics tools .
2. Open-Source
3. Cross-Platform, works on multiple operating systems like Windows, macOS, Linux, and even mobile platforms like Android. Other APIs like DirectX are platform-specific (e.g., Windows only).
4. Wide hardware compatibility, supported by most GPUs, from desktop to mobile devices.

# Using OpenGL Across Different Programming Languages

OpenGL itself is a **C**-based API, but you can use it in many programming languages through libraries such as:

- **C++:** Commonly used due to its performance and flexibility.
- **Python:** Libraries like PyOpenGL provide OpenGL support.
- **Java:** Libraries like JOGL (Java OpenGL).

# Why OpenGL is Cross-Platform

**OpenGL's Design Philosophy:**

OpenGL focuses exclusively on rendering 2D and 3D graphics. It does **not** manage windows, input, or events, making it **independent** of operating systems.

Developers can use external special platform libraries to handle operating system tasks. For example : **GLUT , GLEW**

**GLUT (OpenGL Utility Toolkit) :** managing OpenGL settings, windows, and user input. Header file : **GL/glut.h**

**GLEW (OpenGL Extension Wrangler) :** removes operating system dependencies.

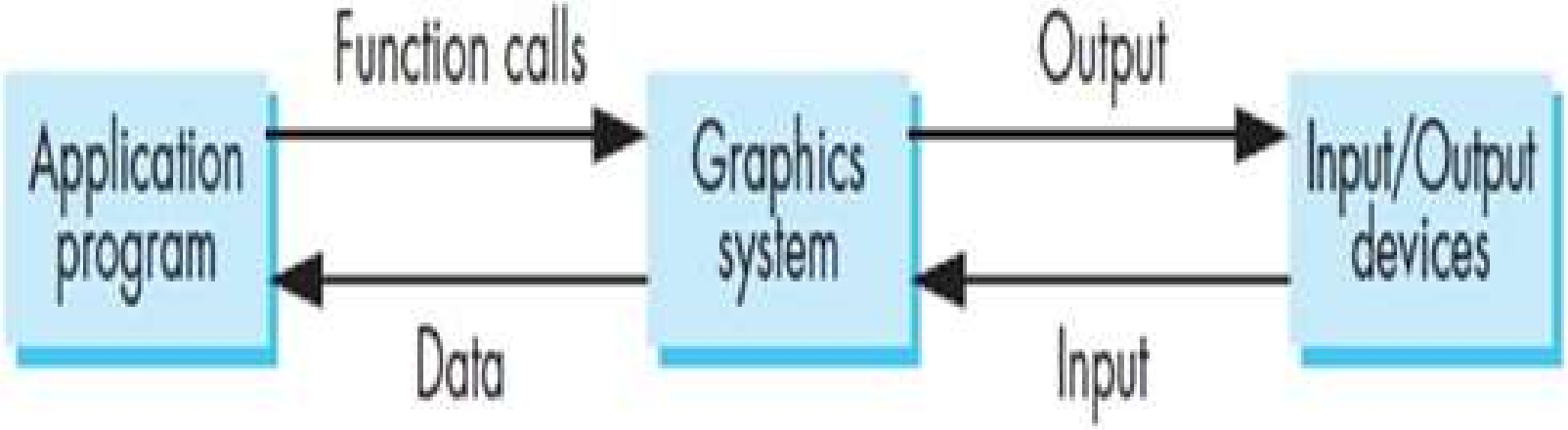
# What is included in GLUT

| Library | Description | Header file | Example functions |
|---|---|---|---|
| GL (OpenGL) | The core graphics library | **GL/gl.h** | glClear()<br>glColor3ub() |
| GLU (OpenGL Utility Library) | A utility library that simplifies tasks like setting up **projection** matrices. | **GL/glu.h** | gluPerspective() |

```
#include <GL/glut.h>
```

To keep your GLUT programs portable, simply include *glut.h*, as it automatically includes *gl.h* and *glu.h* for you. There's no need to include all three files separately.

# Graphics System as a Blackbox




- **Inputs are :**
  - function calls from an application program.
  - measurements from input devices, such as the mouse and keyboard.
  - messages from the outputs.
- **Output** is graphics sent to output devices.

**Graphics Functions categories**

- Primitive functions
- Attribute functions
- Viewing functions
- Transformation functions
- Input functions
- Control functions
- Query functions

# Primitive functions

Define basic shapes that serve as building blocks for rendering objects.

These shapes are called primitives, and they include: Points, Lines, Triangles, Polygons.

Other complicated objects are not directly supported by OpenGL, there are other libraries that provide the necessary code.

Example :

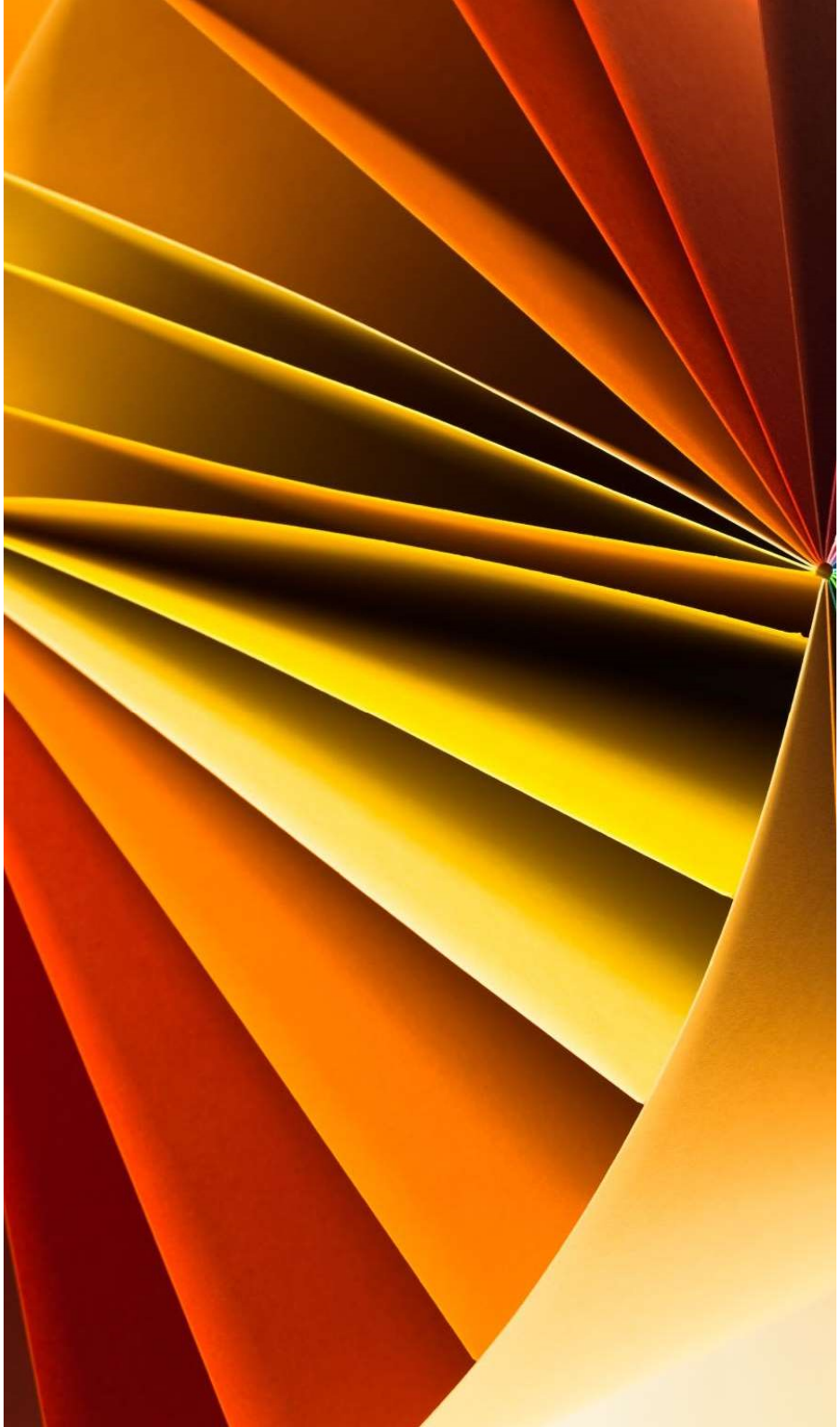
```
GL_POINTS , GL_LINES

glBegin(GL_QUADS);
```

# Attribute functions

Define visual properties that affect how objects are rendered. They control aspects such as:

- Colors: `glColor3f()`
- Materials : `glMaterialfv()`
- Lighting : `glLightfv()`
- Textures: `glTexParameter()`



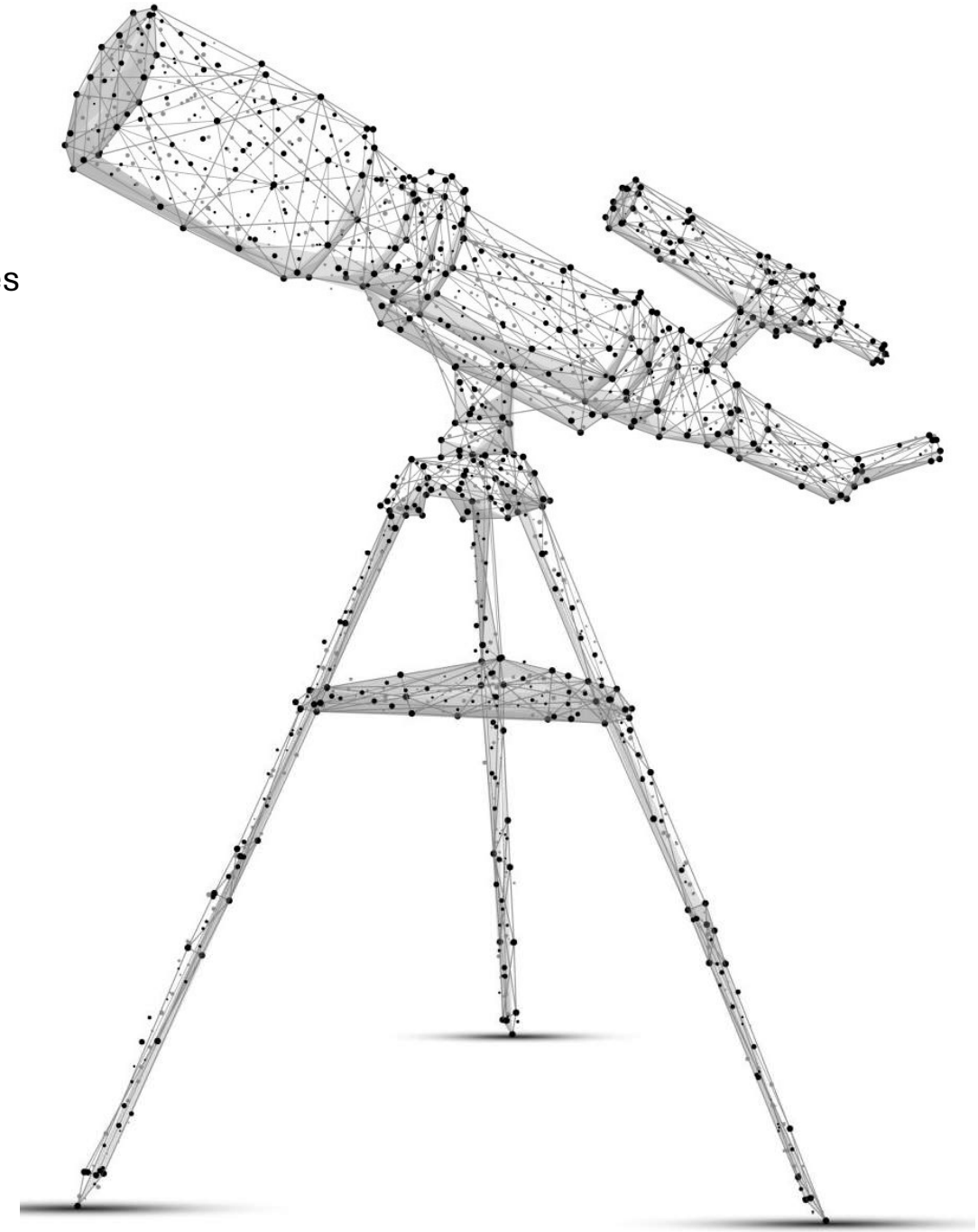
# Viewing functions

OpenGL does not provide any viewing functions but relies on the use of transformations in the shaders to provide the desired view.

- **3D Camera:**
  - Simulate camera movement using transformations (translation, rotation).
  - Example: `gluLookAt()`
- **Projections:**
  - **Perspective Projection:** Realistic depth, objects farther away appear smaller.

    Example: `gluPerspective()`
  - **Orthographic Projection:** Maintains size regardless of distance.

    Example: `glOrtho()` , `gluOrtho2D()`



# Transformation functions

Used to manipulate the position, size, and orientation of objects in 3D space.

They modify objects through:

- **Translation**: Moving objects to a new position.

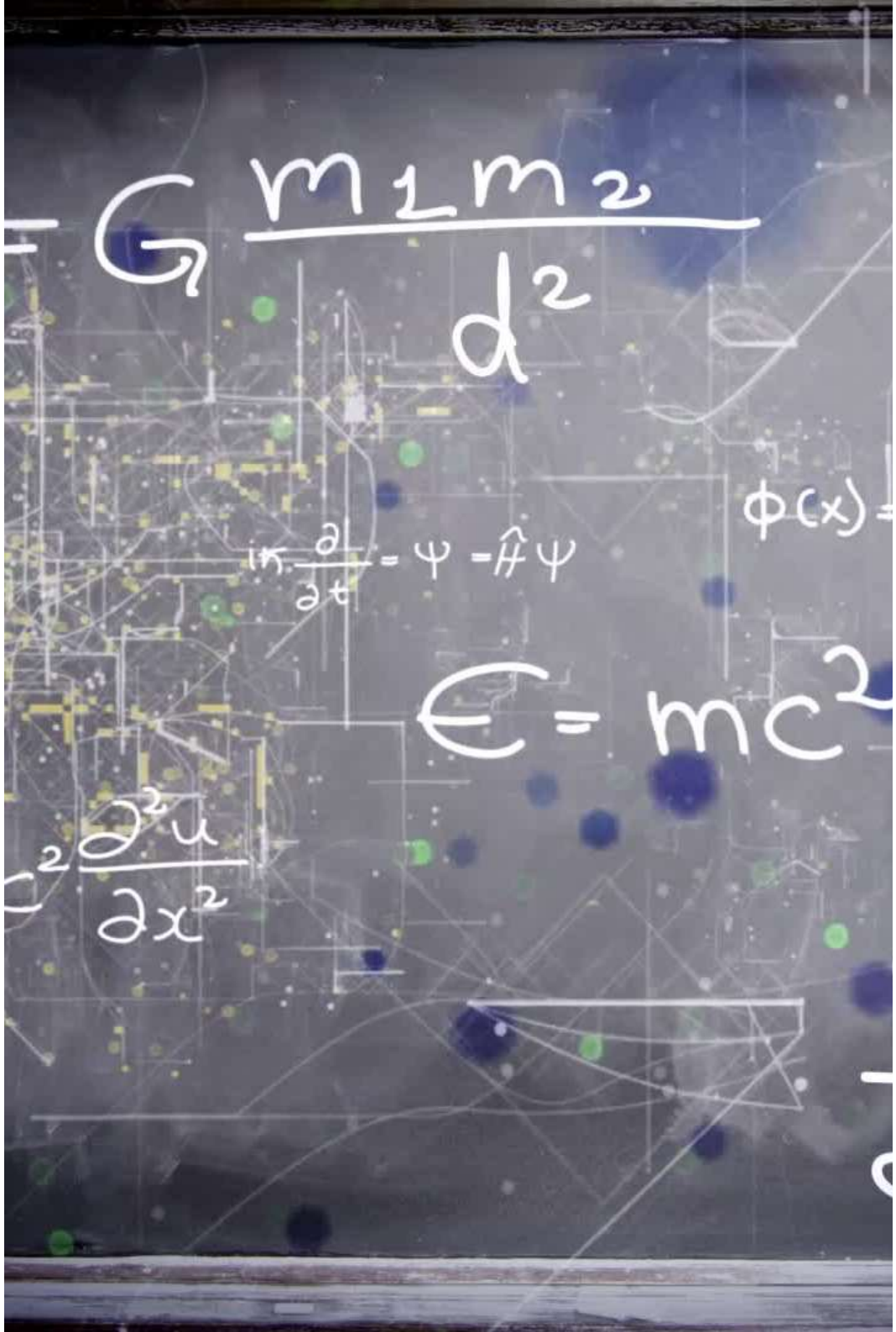
  Example: `glTranslatef()`

- **Rotation**: Rotating objects around an axis.

  Example: `glRotatef()`

- **Scaling**: Changing the size of objects.

  Example: `glScalef()`



# Input functions

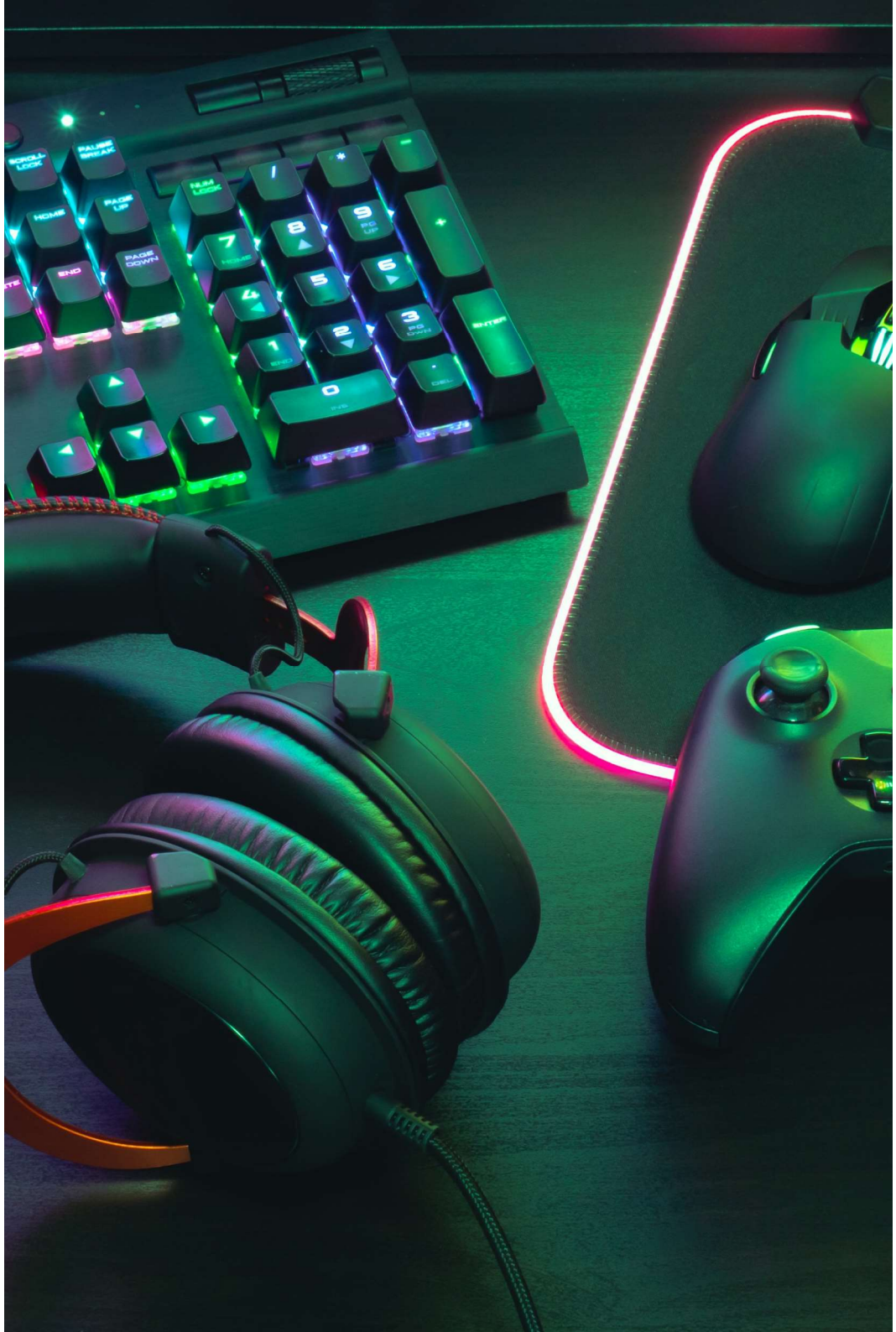
OpenGL itself does not handle input directly. Input functions are provided by external libraries like GLUT to process user interactions such as:

- Mouse clicks and movements.

  Example: `glutMouseFunc()`
- Keyboard presses.

  Example: `glutKeyboardFunc()`
- Window events.

  Example: `glutReshapeFunc()`



# Control functions

Manage the state of the graphics pipeline and the behaviour of rendering. They influence aspects like:

- Enabling/disabling specific features.

  Example□□□

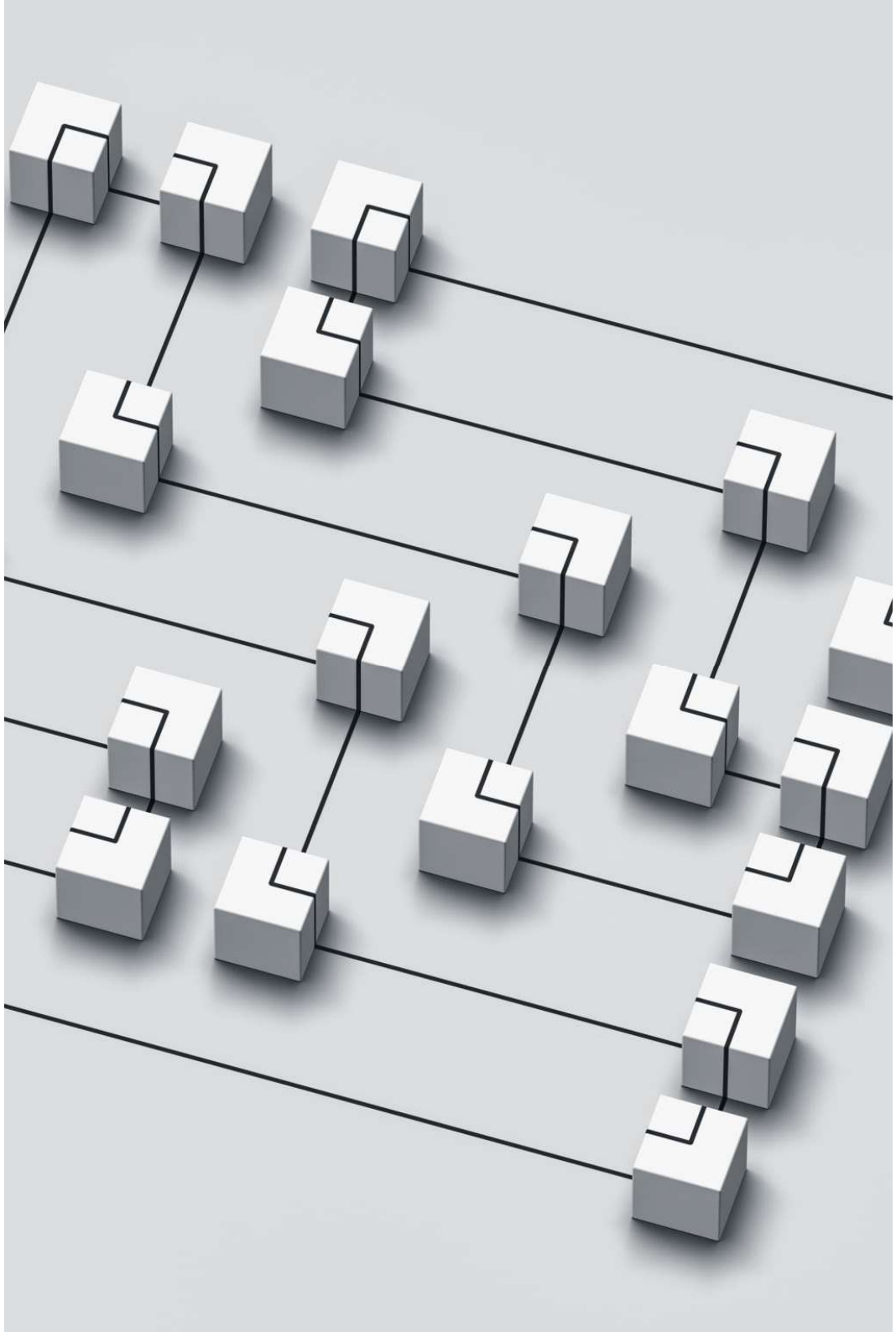
`glEnable() / glDisable()`

- Configuring how OpenGL processes data (e.g., depth)

  Example: `glDepthFunc()`

- Controlling rendering modes and options.

  Example : `glFlush()`



# Query functions

Query functions in OpenGL allow the program to retrieve information about the current **state** or **capabilities** of the OpenGL context.

Example: `glIsEnabled()` , `glGetString()`